MANUALE DEL PROGRAMMA CHAMP

Champ funziona sui microcomputer: Commodore 64 e VIC 20 (32 K), BBC Modello B e Sinclair Spectrum (48K) ✻ Il package comprende un Assembler per linguaggio ASSEMBLY 6502/6510 o Z80, un editor ed un monitor/debugger/disassembler ✻ Questi programmi costituiscono un valido strumento di lavoro per chi desideri programmare in codice macchina

Caricamento del programma

Sul COMMODORE 64: posizionare la cassetta al giro 70 e premere contemporaneamente [SHIFT] e [RUN/STOP]. In alternativa, LOAD "CHAMP" e attendere il caricamento, quindi digitare RUN

Sul VIC 20: premere assieme [SHIFT] e [RUN/STOP]

Sul BBC: premere CHAIN ""

Sullo Spectrum: regolare al massimo volume e toni alti, far girare la cassetta e bloccarla a circa 140 giri, digitare LOAD "CHAMP" e premere [ENTER]. Avviare il registratore per caricare la prima parte. Terminata questa fase, il registratore si blocca e si spegne: digitare RUN e premere [PLAY]

L'avvio del programma Champ è automatico, pertanto occorre soltanto attendere, dopo averlo caricato, che appaia sullo schermo il messaggio di Copyright. A questo punto, si rimuova la cassetta dal registratore, sostituendola con un'altra vergine, sulla quale memorizzare i programmi generati mediante il Champ.

Oltre al messaggio di Copyright appaiono altre informazioni, riguardanti le locazioni di memoria occupate da Champ. È consigliabile annotarsi questi valori, la cui utilità è chiarita nella didascalia a fianco. Fatto ciò, si prema il tasto [ESC] per proseguire. Sullo schermo apparirà:

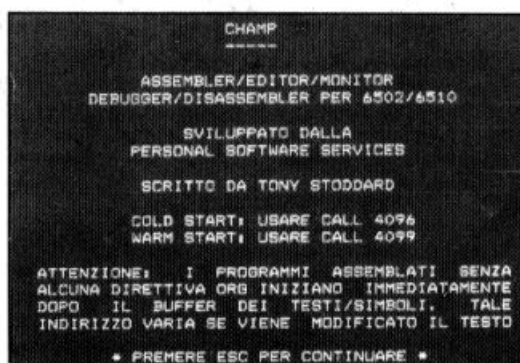
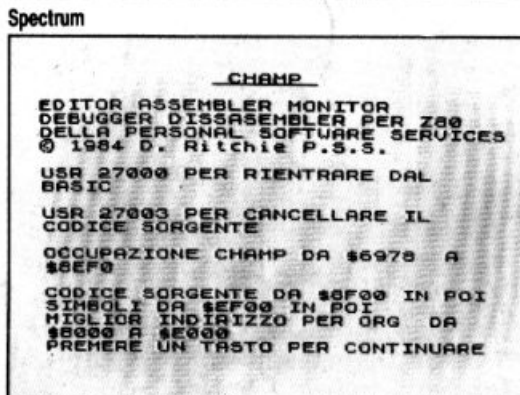
A questo punto, il computer è in attesa che venga scritto un programma in Assembly, ma prima di procedere, si osservino i due programmi riportati più avanti nel manuale e si scelga quello adatto al proprio computer (la versione 6502 è adatta al BBC ed ai Commodore, mentre quella Z80 è per lo Spectrum).

La prima cosa che si nota, nei programmi, è una certa abbondanza di punti e virgola (;). Nel linguaggio Assembly, questo carattere è l'equivalente della REM in BASIC, pertanto tutte le linee che iniziano con un punto e virgola servono soltanto come linee di COMMENTO. La presenza di commenti facilita sempre la comprensione di un programma e nel nostro esempio abbiamo

NOTAZIONE DEI TASTI

Nel manuale, una (o più lettere) tra parentesi quadre, ad esempio [A] o [RET] significa: "il tasto che riporta questo (questi) caratteri". Ecco il significato di ciascun simbolo:

SIMBOLO	COMMODORE	BBC	SPECTRUM
[RET]	Tasto [RETURN]	Tasto [RETURN]	Tasto [ENTER]
[ESC]	Tasto [←]	Tasto [ESCAPE]	[CAPS SHIFT] + [1]
[CRSRR]	[Cursore a destra]	[Cursore a destra]	[CAPS SHIFT] + [8]
[CRSRL]	[Cursore a sinistra]	[Cursore a sinistra]	[CAPS SHIFT] + [5]
[↑]	[Cursore in alto]	[Cursore in alto]	[CAPS SHIFT] + [7]
[↓]	[Cursore in basso]	[Cursore in basso]	[CAPS SHIFT] + [6]
[CTRL]	Tasto [CTRL]	Tasto [CTRL]	[CAPS SHIFT]
[SP]	Carattere spazio	Carattere spazio	Carattere spazio

Commodore

BBC Micro

Spectrum

Lo schermo iniziale

Avviando CHAMP compaiono diverse scritte di presentazione. In particolare appaiono gli indirizzi di "Warm start" e di "Cold start". Ambedue gli indirizzi servono per accedere nuovamente a CHAMP dopo aver lavorato in BASIC, mediante PRINT USR indirizzo, SYS indirizzo, oppure CALL indirizzo (dove "indirizzo" è uno dei due numeri indicati).

La differenza tra "warm start" e "cold start" è che il primo consente di riprendere il lavoro lasciato in sospeso con CHAMP senza cancellare eventuali listati già immessi, mentre il secondo azzera tutte le variabili di servizio di CHAMP, preparandolo ad un nuovo lavoro.

inserito alcune spiegazioni ed esempi sulle funzioni svolte, confrontandole con l'equivalente BASIC.
Ogni linea che non inizi con un punto e virgola rappresenta una linea di programma. La prima di queste contiene la direttiva ORG $CØØØ, che stabilisce l'indirizzo di memoria dal quale far partire il programma in codice macchina. ORG sta per ORIGINE e $CØØØ è l'indirizzo in esadecimale (in quanto preceduto da $) equivalente a 49.152 in decimale. In BASIC non è necessario stabilire un indirizzo iniziale per i programmi, poiché di questo si occupa l'interprete BASIC, ma in codice macchina siamo a diretto contatto con il computer e dobbiamo quindi comunicargli anche questo genere di informazioni.
Infatti, occorre anche stabilire dove, nella memoria, conservare le variabili. Nei programmi più complessi può risultare comodo riunire tutte queste informazioni in un unico blocco, magari all'inizio del programma, mentre nei programmi più semplici la definizione di una variabile può anche essere collocata accanto alle poche istruzioni che la riguardano. Nel nostro esempio, le due linee successive comunicano all'assembler che intendiamo usare due variabili, I e J, collocate subito all'inizio del programma. Siccome non ci servono valori maggiori di 256, per ciascuna variabile è sufficiente un solo byte e usiamo quindi la direttiva DB (Define Byte). Se volessimo usare numeri maggiori, potremmo impiegare la direttiva DW (Define Word) che riserva due byte (= word). A puro titolo d'esempio, in BASIC una variabile occupa generalmente cinque byte. Sia la DB che la DW, dunque, comunicano al computer che intendiamo usare alcuni byte come deposito per le variabili, associando ad esse i nomi I e J.
Le linee di programma finora viste contengono soltanto pseudo-operatori (ORG, DB, ecc.), così chiamati perché non generano effettivamente alcun codice macchina, ma servono soltanto per comunicare all'assembler alcune informazioni.
Il resto del programma (a parte i commenti, ovviamente) è composto invece da istruzioni che producono codice macchina, successivamente eseguibile dal microprocessore. Tra queste istruzioni incontriamo: carica il registro A, esamina un "flag", "salta" ad un nuovo indirizzo, ecc.
Adesso possiamo trascrivere il programma, come descritto nel capitolo successivo.

Modi di funzionamento

Modo <ASSEMBLE>

viene usato una volta immesso un programma in linguaggio Assembly, per convertirlo in codice macchina eseguibile

Modo <INSERT>

viene usato per immettere il listato "sorgente" del programma

Modo <EDIT>

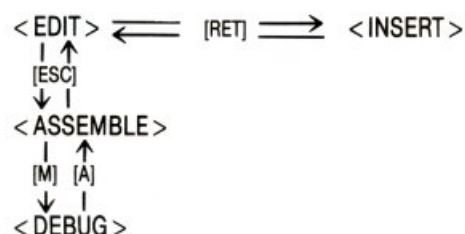
serve per apportare modifiche ad un listato precedentemente immesso

Modo <DEBUG>

permette di ispezionare il codice macchina generato mediante le operazioni precedenti, o di verificare il corretto funzionamento di un programma

Nel Modo <ASSEMBLE> e nel Modo <DEBUG> l'operatore impartisce dei comandi (descritti più avanti) per ottenere particolari funzioni. Nei Modi <INSERT> ed <EDIT>, invece, l'operatore digita dei caratteri, che formeranno il listato del programma o, comunque, interviene per modificare il testo immesso.

Il seguente diagramma mostra come passare da un Modo all'altro:

```
<EDIT>  <===  [RET]  ===>  <INSERT>
 | ↑
[ESC]
 ↓ |
<ASSEMBLE>
 |   ↑
[M] [A]
 ↓   |
<DEBUG>
```

Trascrizione ed esecuzione dei programmi sui Commodore e sul BBC

Se Champ è stato appena caricato in memoria ed avviato, non esiste ancora alcun programma da "assemblare", quindi il modo <ASSEMBLE> non può essere usato, per il momento. Si passi al Modo <EDIT> (premendo ESCAPE sul BBC, ← sui Commodore): sulla linea di edit apparirà un cursore lampeggiante. La prima cosa da fare è inserire un commento per descrivere il programma: si digiti un punto e virgola, seguito dal nome del programma e da opportune informazioni. La linea va terminata, prima di giungere al margine destro della linea, premendo il tasto RETURN. La linea appena inserita si sposta automaticamente verso l'alto, ed il cursore lampeggiante si posiziona all'inizio di una nuova linea. Sul fondo dello schermo appare la scritta <INSERT>, in quanto stiamo "inserendo" istruzioni. Se commettessimo un errore di battitura prima di aver premuto RETURN, si possono usare i tasti di movimento del cursore per provvedere alla correzione. Se l'errore viene rilevato dopo la pressione di RETURN si possono ancora apportare correzioni, come vedremo tra poco.
Per rendere più chiaro il listato, si possono inserire delle linee di commento vuote (che contengono il solo punto e virgola).
Se, terminata una linea, si preme ancora una volta RETURN si entra nel modo <EDIT>, che consente di usare i tasti di movimento del cursore per spostarsi avanti e indietro sul testo. Si possono apportare correzioni al testo, ma non si deve premere RETURN. Al termine della correzione, riportare il cursore in fondo al testo, pronti per immettere una nuova linea.
Digitando uno spazio, nella prima posizione, si nota che il cursore si sposta automaticamente al campo succes-

sivo, dove adesso possiamo scrivere ORG. Di nuovo, digitando uno spazio, il cursore si posiziona sul campo degli operandi, dove scriviamo $CØØØ, seguito da RETURN. Tutte le istruzioni (eccetto i commenti) vengono immesse seguendo questo criterio.

Se desideriamo inserire un'etichetta (label), ciò va fatto nel primo campo (si veda NEXTI o NEXTJ nell'esempio). Anche in questo caso, digitando uno spazio si provoca lo spostamento del cursore sul campo successivo.

Alcuni errori di battitura vengono immediatamente riconosciuti da CHAMP non appena si preme RETURN al termine della linea. In tal caso appare una scritta di errore nell'apposita linea dello schermo. Possibili errori, in questa fase, sono quelli che concernono i tre campi, ossia: quello delle LABEL (etichette), quello delle ISTRUZIONI e quello degli OPERANDI. Occorre allora controllare il campo relativo per individuare l'errore.

Terminata l'immissione del listato, si prema ancora una volta RETURN per tornare al modo <EDIT>. Usando i tasti di movimento del cursore si esamini l'intero listato confrontandolo con l'originale. Se tutto è a posto, si può passare al modo <ASSEMBLE> (tasto ESCAPE sul BBC, ← sui Commodore). Abbiamo adesso la possibilità di memorizzare sul nastro il listato (operazione molto utile per non doverlo riscrivere nuovamente in altre occasioni). Per far ciò sul BBC e sui Commodore basta premere il tasto W. Il listato potrà essere riletto, in occasioni successive, usando il tasto L.

Una volta memorizzato il listato, possiamo assemblarlo. Premendo il tasto A sul fondo dello schermo appare la scritta ASSEMBLE = >. CHAMP attende adesso che selezioniamo il numero corrispondente al tipo di assemblaggio desiderato (vedere tabella delle opzioni). Per esempio si prema il tasto 3, seguito da RETURN.

Se tutto procede regolarmente, sullo schermo viene visualizzato il listato originale corredato da una serie di numeri (esadecimali) nella parte sinistra. Questi numeri costituiscono l'indirizzo di memoria occupato da ogni istruzione o dato del programma ed il codice macchina depositato in tale indirizzo. Si noti che le linee di commento non influiscono sulla numerazione degli indirizzi.

Al termine del listato viene visualizzata una tabella dei simboli usati (Symbol Table): essa contiene tutte le "etichette" usate nel programma ed è molto utile per individuare una particolare etichetta all'interno del programma.

Avendo assemblato con successo il programma, possiamo esaminare le locazioni di memoria nelle quali esso è contenuto. Per far ciò si usa il sottoprogramma di monitor. Si prema il tasto M e comparirà la scritta <DEBUG>.

L'indirizzo iniziale del programma non è $CØØØ, ma bensì $CØØ2, poiché le prime due locazioni sono occupate dalle variabili I e J. Pertanto occorre impartire il comando Q (per disassemblare), seguito dall'indirizzo CØØ2 (senza il simbolo $).

Appena si preme RETURN sullo schermo appare un listato simile a quello da noi digitato, ma senza commenti, né nomi di variabili, né etichette, né pseudo-op. Si ricordi, infatti, che non viene prodotto alcun codice macchina per queste parti del programma. Per proseguire l'esame del contenuto della memoria basta premere qualsiasi tasto, mentre per interrompere questa operazione si prema ESCAPE sul BBC, ← sui Commodore. Per rivedere il disassemblato, occorre impartire di nuovo il comando QCØØ2.

Qualora durante questa fase il listato non dovesse assomigliare affatto a quello originale, si provi ad assemblare nuovamente il programma, prestando particolare attenzione che non vengano segnalati errori da parte dell'Assembler.

Dal "SORGENTE" al "DISASSEMBLATO" versione 6502

Qui a fianco è riportato il listato "sorgente" del programma dimostrativo per i Commodore e il BBC che deve essere trascritto con cura ed attenzione onde evitare errori.

Le varie istruzioni del "sorgente" vengono interpretate dall'Assembler e tradotte in codice macchina (vedere le prime due colonne del "disassemblato").

Successivamente, il contenuto della memoria può essere esaminato e riconvertito (nel modo <DEBUG>) in istruzioni mnemoniche.

LISTATO DISASSEMBLATO - 6502

```
CØØ2    A264     LDX    #$64
CØØ4    8EØØCØ   STX    $CØØØ
CØØ7    A2FF     LDX    #$FF
CØØ9    8EØ1CØ   STX    $CØØ1
CØØC    AEØ1CØ   LDX    $CØØ1
CØØF    CA       DEX
CØ1Ø    DØF9     BNE    $CØØ9
CØ12    AEØØCØ   LDX    $CØØØ
CØ15    CA       DEX
CØ16    DØEE     BNE    $CØØ4
CØ18    6Ø       RTS
```

```
;
;               ESEMPIO DI PROGRAMMA
;               VERSIONE 6502
;
                ORG $CØØØ
;
;               VARIABILI
;
I               DB          Ø
J               DB          Ø
;
;               PROGRAMMA
;
;1Ø FOR I = 1ØØ TO 1 STEP -1
                LDX         #64
NEXTI           STX         I
;2Ø FOR J = 255 TO 1 STEP -1
                LDX         #$FF
NEXTJ           STX         J
;3Ø NEXT J
                LDX         J
                DEX
                BNE         NEXTJ
;4Ø NEXT I
                LDX         I
                DEX
                BNE         NEXTI
;5Ø RETURN (AL BASIC OPPURE AL CHAMP)
                RTS
```

COMANDI NEL MODO <EDIT>

Nel modo <EDIT> viene visualizzato il testo del programma, con il cursore posizionato sulla riga di correzione e la scritta <EDIT> sulla riga dei comandi. Il testo può essere modificato o cancellato (usando [DEL] o [SPAZIO]). Il tasto [RET] provoca una verifica sulla sintassi ed il formato della linea appena modificata. Se vi sono errori, ciò viene segnalato e la linea rimane sulla riga di correzione, altrimenti essa va a far parte del listato del programma ed il funzionamento passa dal modo <EDIT> ad <INSERT>. Usando il tasto [RET] si può passare dall'uno all'altro di questi due modi, mentre il tasto [ESC] permette il passaggio da <EDIT> ad <ASSEMBLE> e viceversa.

Per muovere il testo, purché la linea attualmente nella riga di correzione sia corretta, si possono adoperare i seguenti tasti. Se viene modificata una linea, ed essa risulta priva di errori, allora i tasti qui descritti hanno l'effetto di inserire la nuova linea nel corpo del testo, senza che sia necessario cambiare il modo di funzionamento.

TASTO	EFFETTO
[↑]	Sposta il testo di una riga in su
[↓]	Sposta il testo di una riga in giù
[CTRL]+[U]	Sposta il testo di una pagina in su
[CTRL]+[D]	Sposta il testo di una pagina in giù
[CTRL]+[T]	Visualizza l'inizio del testo
[CTRL]+[B]	Visualizza la fine del testo
[CTRL]+[Z]	Cancella il contenuto della riga di correzione
[ESC]	Passa al modo <ASSEMBLE>
[RET]	Passa al modo <INSERT>

N.B. I comandi di posizionamento del testo producono gli stessi effetti anche nel modo <ASSEMBLE> (non si usi, però, il tasto [CTRL]). Nel modo <ASSEMBLE>, pertanto, il tasto [U] muove il testo di una pagina in su.

COMANDI NEL MODO <INSERT>

Nel modo <INSERT> si immette il testo del programma da assemblare. Sulla riga dei comandi compare la scritta <INSERT>, mentre su quella di correzione appare, lampeggiante, il cursore. L'intero schermo è suddiviso in tre zone colorate, che corrispondono, rispettivamente, al campo delle Etichette, a quello delle Istruzioni ed a quello degli Operandi del programma in Assembly.

Campo delle Etichette

Per etichette s'intende una qualsiasi stringa alfanumerica lunga sei caratteri, il primo dei quali deve essere una lettera. Non occorre far seguire l'etichetta dal simbolo dei due punti, né da qualsiasi altro carattere delimitatore.

Campo delle Istruzioni

Le istruzioni appartengono ai linguaggi ASSEMBLY specificati dalla MOS Technology per il 6502 e dalla Zilog per lo Z80. Esse possono essere composte da due, tre o quattro caratteri e devono iniziare dalla prima posizione nel Campo.

Campo degli Operandi

Gli operandi possono essere costanti esadecimali (purché precedute dal simbolo $), etichette, simboli o anche espressioni contenenti due operandi separati da + o —.

Non sono consentite costanti espresse in decimale, né in binario. Il formato degli operandi deve corrispondere, per il resto, a quanto specificato dalla MOS Technology e dalla Zilog.

L'immissione del testo, nel modo <INSERT>, è governata da un "formattamento per campi": ciò significa che è impossibile immettere un'etichetta lunga più di sei caratteri od un'istruzione lunga più di cinque. Infatti, tentando di superare questi limiti, oppure premendo uno [SPAZIO], il cursore si sposta automaticamente nel campo successivo.

I tasti [CRSRR], [CRSRL] e [DEL] funzionano normalmente nel modo <INSERT>, salvo il fatto che [DEL] cancella il carattere sotto al cursore, anziché quello alla sua sinistra.

Premendo il tasto [RET], la linea appena digitata viene sottoposta ad un controllo formale. Se vengono rilevati errori, ciò viene segnalato nell'apposita riga dello schermo. Se la linea è priva di errori, allora la linea è aggiunta al listato e la riga di correzione viene "ripulita" per consentire l'immissione di una nuova linea. Battendo il tasto [RET] senza aver scritto niente nella riga di correzione, si provoca, alternativamente, il passaggio dal modo <EDIT> a quello <INSERT> e viceversa.

COMANDI DEL MODO <DEBUG>

Il modo <DEBUG> comprende tre diversi tipi di funzione:

Monitor della memoria
Consente di ispezionare e modificare il contenuto di ogni singola locazione di memoria del computer.

Disassemblatore
Consente di interpretare il contenuto delle varie locazioni di memoria sotto forma di codici mnemonici del linguaggio ASSEMBLY. In altre parole, permette di risalire alle istruzioni che formano il programma.

Debugger
Consente di eseguire il programma sotto prova, ma sfruttando speciali meccanismi per l'individuazione di eventuali errori.

Nel modo <DEBUG> lo schermo non risulta più suddiviso in campi: al contrario, esso è inizialmente vuoto, con la sola scritta <DEBUG> in alto a sinistra. Tutti i valori immessi sono considerati esadecimali (anche senza impiegare il simbolo $). Tuttavia, il comando H può gestire costanti numeriche in forma decimale.

ABBREVIAZIONI	
addr	qualsiasi indirizzo esadecimale
saddr	indirizzo di partenza di un blocco di memoria
faddr	indirizzo finale di un blocco di memoria
hx	un valore hex (hx < $FF)
regname	il nome di un registro della CPU (vedere oltre)
expr	una qualsiasi espressione aritmetica con uno o due operandi. Gli operandi possono essere costanti decimali, esadecimali (se viene usato il prefisso $) o simboli consentiti. Le operazioni consentite sono: + e —.
bystr	una stringa di valori hex, separati da spazi
chstr	una stringa di caratteri (così come appare, senza separatori)

COMANDO	EFFETTO
@ addr	Per ogni locazione, a partire dall'indirizzo specificato, ne viene visualizzato il contenuto in esadecimale ed il corrispondente carattere ASCII. Usare il tasto [RET] per passare alla locazione successiva, oppure [ESC] per tornare a livello comandi. Per modificare il contenuto della locazione in esame, basta digitarlo
A	Riporta al modo <ASSEMBLE>
D addr	Il contenuto della memoria, in esadecimale, viene visualizzato ad intere pagine per volta. Per passare alla pagina successiva premere qualsiasi tasto, salvo [ESC], che riporta a livello comandi.
F saddr faddr hx	Ciascun byte della memoria compresa tra saddr e faddr viene riempito col valore hx
M daddr saddr faddr	Il blocco di memoria compreso tra saddr e faddr viene ricopiato nella zona di memoria a partire dalla locazione daddr
Q addr	Le locazioni di memoria, a partire da addr, vengono disassemblate. [RET] prosegue con le locazioni successive, mentre [ESC] riporta a livello comandi
G addr	Viene eseguito il programma che inizia alla locazione addr (uscendo dal programma CHAMP)
C addr	Viene eseguito il programma che inizia alla locazione addr (tornando, alla fine, al programma CHAMP)
Bn = addr	Serve per attivare una *breakpoint*, ossia una sospensione nell'esecuzione del programma, durante la quale è possibile esaminare il contenuto della memoria. Il numero della breakpoint (n) può variare da 1 a 8. Per proseguire l'esecuzione, premere [C] [RET]
En	Disattiva la breakpoint numero n.
T	Visualizza gli indirizzi delle breakpoint al momento attive
R regname	Visualizza il contenuto del registro della CPU specificato, permettendo di modificarlo
J addr	Esegue il programma "passo passo", a partire dall'indirizzo specificato. Ad ogni passo, viene mostrato il contenuto dei registri della CPU. Il tasto [J] permette di proseguire, mentre [ESC] provoca il ritorno a livello comandi
H expr	Visualizza l'equivalente decimale, esadecimale e binario di expr
S bystr	Ispeziona la memoria, a partire dalla locazione $ØØØ, alla ricerca di ogni occorrenza di bystr. Sullo schermo appare la scritta "Searching", per avvertire che la ricerca è in corso. Quando la sequenza è trovata, ne viene mostrato l'indirizzo. Usare [RET] per continuare nella ricerca, oppure [ESC] per tornare a livello comandi
N chstr	Come il comando precedente, ma per sequenze di caratteri alfanumerici
W	Per scrivere, leggere o verificare un programma in codice macchina sull'unità a nastri. Vedere tabella BASIC

ABBREVIAZIO DEI REGISTR

6502

A = Accumulatore; X, Y = registri X, Y; P = registro di condizione della CPU; SP = Stack Pointer

IL MODO <

IL FORMATO DEI COMANDI

Find = >stringa [RET]
cerca, partendo dall'inizio del testo del programma, la stringa specificata nel comando.

Next = >stringa [RET]
Come il comando precedente, ma la ricerca inizia dalla linea successiva a quella che si trova nella riga di correzione.

Find = >[RET] e **Next = >[RET]**
In questo caso prosegue la ricerca della stringa specificata nell'ultimo comando F o N. Durante la ricerca, nella riga riservata agli errori, compare il messaggio "Searching" (sto cercando). Se la ricerca ha successo, la linea di programma trovata viene visualizzata nella riga di correzione, altrimenti appare, nella stessa riga, l'ultima linea del programma.

Load = >Save = >Verify = >
Ciascuno dei tre comandi deve essere seguito dal nome di un file (non occorre racchiuderlo tra doppi apici). Ovviamente il nome deve attenersi al formato normalmente usato sul proprio computer.

Print = >espressione [RET]
Questo comando visualizza, nella riga riservata agli errori, il risultato hex dell'espressione specificata. Ad esempio: Print = >$F8-$C1 $37
Si possono adoperare i simboli usati nel testo del programma, ma nell'espressione è consentito indicare solo un operatore (+ o —) alla volta.

Quit = >[Y]
Questo comando serve per uscire dal programma CHAMP a lavoro finito, ma soltanto se si digita, per conferma, una Y (Yes). La pressione di qualsiasi altro tasto equivale a negare tale conferma.

[M]
La pressione del tasto M consente di entrare nel modo <DEBUG>. Per ritornare da questo al modo <ASSEMBLE> digitare: [A] [RET].

[ESC]
Il tasto [ESC] permette di alternare il modo <EDIT> con quello <ASSEMBLE>.

Assemble = >[opzione] [RET]
Questo comando serve ad assemblare il programma secondo le direttive specificate mediante il numero dell'opzione (vedere apposito riquadro).

NI PER I NOMI RI DELLA CPU

Z80

A = Accumulatore; F = registro di condizione della CPU; H,L,B,C,D,E = registri H,...E; SP = Stack Pointer; IX, IY = registri IX ed IY

ASSEMBLE>

COMANDI

TASTO	SCRITTA	FUNZIONE
[F]	Find=>	Cerca una stringa
[N]	Next=>	Cerca una stringa
[L]	Load=>	Legge un listato sorgente
[W]	Write=>	Scrive un listato sorgente
[V]	Verify=>	Verifica un listato sorgente
[P]	Print=>	Visualizza il risultato di un'espressione
[Q]	Quit=>	Per tornare al BASIC
[M]		Per passare al <DEBUG>
[ESC]		Per passare a <EDIT>
[A]	Assemble=>	Per assemblare un programma

Per evitare l'esecuzione involontaria di uno di questi comandi, premere RETURN senza specificare alcun operando.

VARIAZIONI SPECTRUM

TASTO	SCRITTA
[J]	Load=>
[S]	Save=>
[sym. shift]+[R]	Verify=>

FORMATO DELLE ISTRUZIONI

6502

ISTRUZIONE		MODO D'INDIRIZZAMENTO
LDA	#$D4	Immediato
LDA	$3C	Pagina Zero (Diretto)
LDA	$A29Ø	Assoluto (Diretto)
LDA	$31FE,X	Indicizzato assoluto
LDA	$7B,X	Indicizzato Pagina Zero
LDA	($2A,X)	Pre-indicizzato (Indiretto)
LDA	($2A),Y	Post-indicizzato (Indiretto) e implicito

Z80

ISTRUZIONE		MODO D'INDIRIZZAMENTO
LD	A,B	Registro/registro (Diretto)
LD	A,$9F	Immediato
LD	($ED46),A	Assoluto (Diretto)
LD	A,(HL)	Registro (Indiretto)
LD	A,(IY+d)	Indicizzato (Indiretto)
CCF		Implicito

FORMATO DEL LINGUAGGIO ASSEMBLY

PSEUDO-OP	SIGNIFICATO
ORG addr	Origine. Specifica l'indirizzo di memoria dal quale partire per l'assemblaggio del programma. Nella linea di programma che contiene la ORG non possono apparire altre etichette.
EQU	Equivalenza. Serve per assegnare ad un'etichetta (digitata nel Campo Etichette) un valore costante, simbolo o espressione.
DB cost/ chsrt	Assegna un valore (una costante od una serie di caratteri) ad uno o più byte consecutivi nella memoria.
DW cost/ simb	Assegna un valore (una costante od un simbolo) a due byte adiacenti (parola) nel formato basso/alto.
DS cost/ simb	Riserva un numero di byte, pari al valore della costante o del simbolo, per usarli come deposito di dati.

ABBREVIAZIONI	
addr	Un indirizzo hex, preceduto da $
cost	Una costante hex, preceduta dal simbolo $. Se usata con DB, la costante deve essere lunga un byte. Sulla medesima linea si possono usare più DB, ad esempio: DB cost [spazio] DB cost [spazio] ...ecc.
chstr	una sequenza di caratteri (stringa) racchiusa tra apici. Ad esempio: 'AB3%9K1Ø'
simb	qualsiasi operando simbolico

OPZIONI DI ASSEMBLAGGIO

	NUMERO DELL'OPZIONE Ø 1 2 3 4 5 6 7
Visualizza l'intero listato sullo schermo	N S N S N S N S
Carica il codice macchina in memoria	N N S S N N S S
Copia il contenuto dello schermo su stampante	N N N N S S S S
Verifica le etichette, i simboli e la sintassi	S S S S S S S S
Visualizza la tabella dei simboli sullo schermo	S S S S S S S S

S = Funzione attiva
N = Funzione non attiva

Per esempio: **Assemble=> 2 [RET]** serve per assemblare il testo del programma, verificandone la correttezza formale e caricando il codice macchina in memoria, a partire dall'indirizzo specificato nella direttiva ORG. La tabella dei simboli compare sullo schermo, ma il listato completo non appare né sullo schermo, né sulla stampante.

Qualsiasi opzione può essere preceduta da un 1, per ottenere una visualizzazione a doppia interlinea sullo schermo (qualora questo sia abilitato dall'opzione che segue).

Se viene rilevato un errore in fase di assemblaggio, ciò viene segnalato nell'apposita riga dello schermo. Inoltre, l'assemblaggio si interrompe e la linea che contiene l'errore viene visualizzata nella riga di correzione.

BASIC E CODICE MACCHINA

Una volta acquisita una certa pratica si vorranno usare i sottoprogrammi scritti in codice macchina nei programmi BASIC, piuttosto che scrivere interi programmi in codice macchina. Ecco il modo più semplice per ottenere ciò:

1) Si usa CHAMP per sviluppare il sottoprogramma e verificarne il funzionamento.

2) Si memorizza su nastro il listato del sottoprogramma, mediante l'apposito comando impartito nel modo <ASSEMBLE>.

3) Si assembla il programma, depositando il codice in memoria a partire da una locazione abbastanza "alta" nella RAM (consultare il manuale del proprio apparecchio), onde non interferire col sistema operativo.

4) Passando al modo <DEBUG> si trasferisce su nastro la porzione di memoria che contiene il sottoprogramma in codice macchina.

5) Si esce dal programma CHAMP

6) Si scrive il programma BASIC (curandosi di riservare, con opportune istruzioni, una porzione di memoria sufficiente a contenere il sottoprogramma in codice macchina). Inserire nel programma le istruzioni necessarie alla lettura ed al trasferimento in memoria del sottoprogramma.

7) Nei passaggi del programma BASIC, ove è necessario ricorrere al sottoprogramma, si inseriscono opportune istruzioni di "chiamata" (CALL, SYS o USR, a seconda del computer), specificando l'indirizzo di partenza del sottoprogramma.

8) Memorizzare su nastro il programma BASIC, seguendo la normale procedura.

Un esempio del risultato di tutto ciò si può ottenere, una volta usciti da CHAMP e ritornati al BASIC, impartendo il comando LIST. Infatti, CHAMP è composto da un brevissimo programma in BASIC, che, a sua volta, carica in memoria tutta una serie di sottoprogrammi in codice macchina.

MESSAGGI D'ERRORE DI CHAMP

I messaggi d'errore compaiono nell'apposita riga ad essi riservata (eccetto quando si è nel modo <DEBUG>, in cui appare la scritta "ERROR" nella posizione occupata dal cursore).

MESSAGGIO	SPIEGAZIONE
LABEL ERROR	Errore di sintassi o di formato nel Campo delle Etichette
INSTRUCTION ERROR	Errore di sintassi o di formato nel Campo delle Istruzioni
OPERAND ERROR	Errore di sintassi o di formato nel Campo degli Operandi
UNDEFINED LABEL	All'etichetta (o simbolo) visualizzato nella riga di correzione non è stato assegnato un valore. Risulta pertanto non definita
JUMP OUT OF RANGE	L'indirizzo al quale l'istruzione di "salto" relativo (visualizzata nella riga di correzione) è esterno all'arco +128/−127 byte
OVERFLOW	L'assemblaggio del programma, all'indirizzo specificato dalla ORG, provocherebbe la occupazione di porzioni di memoria riservate. Occorre cambiare l'indirizzo dopo la ORG
ERROR	L'operando di un comando impartito nel modo <DEBUG> contiene un errore (valore troppo grande, simbolo errato, indirizzo sbagliato, ecc.)

A questo punto, dopo aver trascritto il programma, averlo assemblato e verificato in memoria, non resta che provarne il funzionamento, sfruttando l'apposita funzione nel modo <DEBUG>. Basta impartire il comando GCØØ2 (equivalente a: esegui dall'indirizzo $CØØ2), seguito da RETURN. Se tutto procede regolarmente, dopo poco ricompare la scritta <DEBUG> ad indicare che il programma è stato eseguito.
Per apprezzare la notevole differenza nei tempi di esecuzione tra il codice macchina ed il BASIC, potremmo scrivere ed eseguire l'equivalente programma in BASIC, ma in tal caso prepariamoci ad un'attesa molto lunga!

Trascrizione ed esecuzione dei programmi sullo Spectrum

Se Champ è stato appena caricato in memoria ed avviato, non esiste ancora alcun programma da "assemblare", quindi il modo <ASSEMBLE> non può essere usato, per il momento. Si passi al Modo <EDIT> (premendo CAPS SHIFT e 1): sulla linea di edit apparirà un cursore lampeggiante. La prima cosa da fare è inserire un commento per descrivere il programma: si digiti un punto e virgola, seguito dal nome del programma e da opportune informazioni. La linea va terminata, prima di giungere al margine destro della linea, premendo il tasto ENTER. La linea appena inserita si sposta automaticamente verso l'alto, ed il cursore lampeggiante si posiziona all'inizio di una nuova linea.
Sul fondo dello schermo appare la scritta <INSERT>, in quanto stiamo "inserendo" istruzioni. Se commettessimo un errore di battitura prima di aver premuto ENTER, si possono usare i tasti di movimento del cursore per provvedere alla correzione. Se l'errore viene rilevato dopo la pressione di ENTER si possono ancora apportare correzioni, come vedremo tra poco.
Per rendere più chiaro il listato, si possono inserire delle linee di commento vuote (che contengono il solo punto e virgola).
Se, terminata una linea, si preme ancora una volta ENTER si entra nel modo <EDIT>, che consente di usare i tasti di movimento del cursore per spostarsi avanti e indietro sul testo del listato. Si possono apportare correzioni al testo, ma non si deve premere ENTER. Al termine della correzione, riportare il cursore in fondo al testo, pronti per immettere una nuova linea.
Digitando uno spazio, nella prima posizione, si nota che il cursore si sposta automaticamente al campo successivo, dove adesso possiamo scrivere ORG. Di nuovo, digitando uno spazio, il cursore si posiziona sul campo degli operandi, dove scriviamo $CØØØ, seguito da ENTER.
Tutte le istruzioni (eccetto i commenti) vengono immesse seguendo questo criterio.
Se desideriamo inserire un'etichetta (label), ciò va fatto nel primo campo (si veda NEXTI o NEXTJ nell'esempio). Anche in questo caso, digitando uno spazio si provoca lo spostamento del cursore sul campo successivo.
Alcuni errori di battitura vengono immediatamente riconosciuti da CHAMP non appena si preme ENTER al termine della linea. In tal caso appare una scritta di errore nell'apposita linea dello schermo. Possibili errori, in questa fase, sono quelli che concernono i tre campi, ossia: quello delle LABEL (etichette), quello delle ISTRUZIONI e quello degli OPERANDI. Occorre allora controllare il campo relativo per individuare l'errore.
Terminata l'immissione del listato, si prema ancora una volta ENTER per tornare al modo <EDIT>. Usando i tasti di movimento del cursore si esamini l'intero listato confrontandolo con l'originale. Se tutto è a posto, si può passare al modo <ASSEMBLE> (tasti CAPS SHIFT e 1). Abbiamo adesso la possibilità di memorizzare sul nastro il listato (operazione molto utile per non doverlo riscrivere nuovamente in altre occasioni). Per far ciò basta premere il tasto S. Il listato potrà essere riletto, in occasioni successive, usando il tasto J.
Una volta memorizzato il listato, possiamo assemblarlo. Premendo il tasto A sul fondo dello schermo appare la scritta ASSEMBLE= >. CHAMP attende adesso che selezioniamo il numero corrispondente al tipo di assemblaggio desiderato (vedere tabella delle opzioni). Per esempio si prema il tasto 3, seguito da ENTER.
Se tutto procede regolarmente, sullo schermo viene visualizzato il listato originale corredato da una serie di numeri (esadecimali) nella parte sinistra. Questi numeri costituiscono l'indirizzo di memoria occupato da ogni istruzione o dato del programma ed il codice macchina depositato in tale indirizzo. Si noti che le linee di commento non influiscono sulla numerazione degli indirizzi.
Al termine del listato viene visualizzata una tabella dei simboli usati (Symbol Table): essa contiene tutte le "etichette" usate nel programma ed è molto utile per individuare una particolare etichetta all'interno del programma.
Avendo assemblato con successo il programma, possiamo esaminare le locazioni di memoria nelle quali esso è contenuto. Per far ciò si usa il sottoprogramma di monitor. Si prema il tasto M e comparirà la scritta <DEBUG>.
L'indirizzo iniziale del programma non è $CØØØ, ma bensì $CØØ2, poiché le prime due locazioni sono occupate dalle variabili I e J. Pertanto occorre impartire il comando Q (per disassemblare), seguito dall'indirizzo CØØ2 (senza il simbolo $).
Appena si preme ENTER sullo schermo appare un listato simile a quello da noi digitato, ma senza commenti, né nomi di variabili, né etichette, né pseudo-op. Si ricordi, infatti, che non viene prodotto alcun codice macchina per queste parti del programma. Per proseguire l'esame del contenuto della memoria basta premere qualsiasi tasto, mentre per interrompere questa operazione si prema CAPS SHIFT e 1. Per rivedere il disassemblato, occorre impartire di nuovo il comando QCØØ2.
Qualora durante questa fase il listato non dovesse assomigliare affatto a quello originale, si provi ad assemblare nuovamente il programma, prestando particolare attenzione che non vengano segnalati errori da parte dell'Assembler.
A questo punto, dopo aver trascritto il programma, averlo assemblato e verificato in memoria, non resta che provarne il funzionamento, sfruttando l'apposita

funzione nel modo <DEBUG>. Basta impartire il comando GCØØ2 (equivalente a: esegui dall'indirizzo $CØØ2), seguito da ENTER. Se tutto procede regolarmente, dopo poco ricompare la scritta <DEBUG> ad indicare che il programma è stato eseguito.

Per apprezzare la notevole differenza nei tempi di esecuzione tra il codice macchina ed il BASIC, potremmo scrivere ed eseguire l'equivalente programma in BASIC, ma in tal caso prepariamoci ad un'attesa molto lunga!

Dal "SORGENTE" al "DISASSEMBLATO" versione Z80

Qui a fianco è riportato il listato "sorgente" del programma dimostrativo per lo Spectrum, che deve essere trascritto con cura ed attenzione onde evitare errori.
Le varie istruzioni del "sorgente" vengono interpretate dall'Assembler e tradotte in codice macchina (vedere le prime due colonne del "disassemblato").
Successivamente, il contenuto della memoria può essere esaminato e riconvertito (nel modo <DEBUG>) in istruzioni mnemoniche.

LISTATO DISASSEMBLATO - Z80

```
CØØ2    3E64     LD    A,$64
CØØ4    32ØØCØ   LD    ($CØØØ),A
CØØ7    3EFF     LD    A,$FF
CØØ9    32Ø1CØ   LD    ($CØØ1),A
CØØC    3AØ1CØ   LD    A,($CØØ1)
CØØF    3D       DEC   A
CØ1Ø    2ØF7     JR    NZ,$CØØ9
CØ12    3AØØCØ   LD    A,($CØØØ)
CØ15    3D       DEC   A
CØ16    2ØEC     JR    NZ,$CØØ4
CØ18    C9       RET
```

```
;
;                ESEMPIO DI PROGRAMMA
;                VERSIONE Z80
;
;
;                ORG $CØØØ
;
;
;                VARIABILI
;
;
I                DB        Ø
J                DB        Ø
;
;
;                PROGRAMMA
;
;
;1Ø FOR I = 1ØØ TO 1 STEP -1
                 LD        A,$64
NEXTI            LD        (I),A
;2Ø FOR J = 255 TO 1 STEP -1
                 LD        A,$FF
NEXTJ            LD        (J),A
;3Ø NEXT J
                 LD        A,(J)
                 DEC       A
                 JR        NZ,NEXTJ
;4Ø NEXT I
                 LD        A,(I)
                 DEC       A
                 JR        NZ,NEXTI
;5Ø RETURN (AL BASIC OPPURE AL CHAMP)
                 RET
```

Osservazioni importanti

Nella stesura dei programmi si rammenti che:

le **ETICHETTE** devono iniziare con una lettera ed essere composte da un massimo di 6 caratteri alfanumerici

le **ISTRUZIONI MNEMONICHE** devono corrispondere allo standard 6502 (oppure Z80)

gli **OPERANDI** devono seguire il formato standard 6502 (oppure Z80). Possono contenere espressioni aritmetiche composte da simboli o costanti esadecimali, compresi il "+" ed il "—", fino a riempire tutto il Campo Operandi

i **COMMENTI** devono iniziare su una nuova riga, usando il carattere ";". Possono riempire, ma non superare, tutta una linea. Su di essi non viene eseguito alcun controllo di sintassi.

Il modo <EDIT> consente di apportare modifiche al contenuto della linea di Edit. Il testo può esser fatto scorrere avanti o indietro sullo schermo, usando i seguenti comandi (per lo Spectrum usare CAPS SHIFT, anziché CTRL):

COMANDO	**EFFETTO**
[↑]	Sposta il testo in alto di una riga
[↓]	Sposta il testo in basso di una riga
[CTRL]+[T]	Sposta all'inizio del testo
[CTRL]+[B]	Sposta alla fine del testo
[CTRL]+[U]	Sposta di una pagina indietro
[CTRL]+[D]	Sposta di una pagina in avanti
[CTRL]+[Z]	Cancella il contenuto della linea di Edit

Questi comandi (senza l'impiego del tasto [CTRL]), hanno un identico effetto nel Modo <ASSEMBLE>, ma non è possibile apportare alcuna modifica al testo del programma.

Se il programma funziona regolarmente, al suo termine ricompare sullo schermo la scritta <DEBUG>, assieme al cursore. Adesso possiamo usare il comando [D] per visualizzare il contenuto della memoria modificato dal programma. Se anche questa verifica è positiva (il programma funziona correttamente), allora possiamo memorizzare il programma su nastro sotto forma di codice "oggetto" (chiamato così per distinguerlo dal "sorgente"), usando il comando "W" in Modo <DEBUG>. Fatto ciò ci possiamo esercitare modificando locazioni di memoria mediante il comando "@", sempre in <DEBUG>. Il miglior metodo per imparare (ed apprezzare) il funzionamento di CHAMP è quello di sperimentare tutti i comandi, verificandone via via gli effetti.
Il peggio che può capitare è di dover "resettare" (o spegnere e riaccendere il computer) e ripartire da zero.

1. Per uscire dal modo <EDIT>

Sullo Spectrum, quando la linea edit è vuota, la pressione di ENTER provoca alternativamente il passaggio dal modo <EDIT> a quello <INSERT>, ma può intervenire un errore. In tal caso, digitare un punto e virgola (;) e premere ENTER. Una volta in <EDIT> o in <ASSEMBLE> si rimuova tale linea di commento.

2. Errore di Label

L'etichetta LABL1 appare definita due volte e ciò rappresenta un errore.

3. Errore d'istruzione

L'istruzione ST non esiste nel linguaggio Assembly dello Spectrum, al quale si riferisce l'esempio, e questo fatto viene prontamente segnalato da CHAMP.

4. Errore di operando

L'esempio mostra un errore nel listato di un Commodore 64: l'istruzione BNE richiede l'uso di un operando e tentare di ometterlo provoca un errore.

5. Errore di assemblaggio

Non tutti gli errori vengono rilevati al momento della trascrizione. In questo caso, la prima linea contiene un errore logico. La pressione di un qualsiasi tasto riporta al modo <ASSEMBLY>.

6. Assemblaggio ultimato

Il programma è stato assemblato senza errori. Avendo scelto l'opzione 11, le linee del listato sono spaziate. Con l'opzione 1 il listato risulta più compatto.

7. Il disassemblatore

Il comando Q, nel modo <DEBUG>, consente di disassemblare (ossia tradurre in codici mnemonici) un blocco di istruzioni in codice macchina. Per tornare al modo <DEBUG> premere ESCAPE, mentre per continuare basta premere qualsiasi tasto.

8. Memory Dump

Il comando D, nel modo <DEBUG> consente di esaminare il contenuto di interi blocchi di memoria, nel formato esadecimale e nel formato ASCII. Premere ESCAPE per tornare al modo <DEBUG> oppure qualsiasi altro tasto per continuare. L'esempio riguarda la versione Z80, in cui vengono mostrati sei byte alla volta. (Nella versione 6502 ne vengono visualizzati otto alla volta).

1

2

3

4

5

6

7

```
AF89 00          NOP

<debug>Q $045D
045D 1EC5        LD    E,$C5
045F CD991E      CALL  $1E99
0462 E1          POP   HL
0463 50          LD    D,B
0464 59          LD    E,C
0465 7A          LD    A,D
0466 B3          OR    E
0467 C8          RET   Z
0468 1B          DEC   DE
0469 C3B503      JP    $03B5
046C CF          RST   8
046D 0A          LD    A,(BC)
046E 89          ADC   A,C
046F 02          LD    (BC),A
0470 D0          RET   NC
0471 12          LD    (DE),A
0472 86          ADD   A,(HL)
0473 89          ADC   A,C
0474 0A          LD    A,(BC)
0475 97          SUB   A
```

8

```
<debug>D $71E2
71E2 D5 71 22 BD 69 FD  Uq"=i}
71E8 71 08 CD 38 7B FD  q.M8{}
71EE 4E 08 2A BD 69 18  N.*=i.
71F4 E5 CD FF 6C 6C 61  eM©lla
71FA 62 65 6C 00 18 20  bel..
7200 CD FF 6C 69 6E 73  M©lins
7206 74 72 75 63 74 69  tructi
720C 6F 6E 00 3E 07 18  on.>..
7212 0D CD FF 6C 6F 70  .M©lop
7218 65 72 61 6E 64 00  erand.
721E 3E 0D FD 77 10 CD  >.}w.M
7224 38 72 4F FD BE 0E  8rO}>.
722A C0 CD CE 6D 01 F6  @MNm.v
7230 FF 09 CD B5 70 0E  ©.M5p.
7236 00 C9 CD FF 6C 20  .IM©l
723C 65 72 72 6F 72 00  error.
7242 21 00 02 11 80 00  !.....
7248 CD B5 03 AF C9 FE  M5./I.
724E 02 20 31 CB E0 CD  . 1K`M
7254 2F 70 1B 0C 0C FE  /p....
725A 24 28 64 FE 30 38  $(.  08
```

Realizzato dalla Personal Software Services
Inserto Redazionale
Stampato in Italia - IGDA Officine Grafiche, Novara - 1985